DICIONÁRIO DO BASIC SINCLAIR

para os micros TK 82C, NE Z8000, TK 83, TK 85, CP 200, RINGO, AS 1000, etc...

Todas as Instruções, Funções, Operadores e Caracteres descritos e exemplificados em ordem alfabética

Delio Santos Lima

DO MESMO AUTOR:

45 PROGRAMAS PRONTOS PARA RODAR EM TK82C e NEZ8000

30 JOGOS PARA TK82 E CP200

CÓDIGO DE MÁQUINA PARA TK E CP200

APLICAÇÕES SÉRIAS PARA TK85 E CP200

SINCLAIR 8KROM - DISASSEMBLY COMPLETO COMENTADO

IMPRESSO POR MICRON ELETRÔNICA
SÃO JOSÉ DOS CAMPOS - SP

DICIONÁRIO DO BASIC SINCLAIR
por Delio Santos Lima

1ª Edição - Julho de 1984
2ª Edição - Setembro de 1984

Impresso,editado e distribuído por
Micron Eletrônica Com.Ind.Ltda.
São José dos Campos - SP

Prefácio

Muitas vezes, já se tendo adquirido razoável conhecimento de Basic, torna-se tedioso e enfadonho utilizar-se de cursos completos, para desvendar os segredos destas ou daquelas instruções ou funções do Basic Sinclair que nos fogem ao controle.

Colocá-las em ordem alfabética, foi a melhor solução para consultas rápidas e esparsas, não sendo recomendado como forma de estudo.

Para tanto, dividi, resumidamente, o Basic Sinclair, nas primeiras páginas deste volume, de forma a esclarecer todas as instruções, comandos, funções e operadores que estão disponíveis aos usuários.

Como não poderia deixar de ser, acrescentei, ainda, a este volume os capítulos : Convertendo outros Basics, Contando os Bytes e Economizando Memória.

Delio Santos Lima
Caixa Postal 100
12.200 São José dos Campos
SP - Brasil

Índice

O que é Basic Sinclair

Desde agosto de 1982, este autor vem empregando os termos "Basic Sinclair" e "Lógica Sinclair".

Muito polemizados, cabe aqui definí-los.

Entende-se por "Basic Sinclair" a linguagem Basic desenvolvida pela Sinclair Research Ltd. em suas versões de 4K e 8KROM, para os microcomputadores ZX80 e ZX81.

Qualquer outra versão de Basic, cujos comandos, instruções , funções, operadores e caracteres forem idênticos a esta, tal que os programas de uma máquina possam ser rodados em outra, sem qualquer alteração , é tida como "COMPATÍVEL" ou a "MESMA".

Já o termo "Lógica Sinclair" especifica os circuitos lógicos característicos ao uso do Basic Sinclair em 4 ou 8K ROM.

Compatibilidade de Soft

Muito tem-se falado de compatibilidade.Compatibilidade de software não existe.É decorrente da compatibilidade de micros. E compatibilidade total também inexiste. Se é inteiramente compatível é o mesmo,é a mesma coisa.

São compatíveis ao Sinclair ZX80 8KROM os seguintes micros nacionais: NEZ8000, TK82C, TK83, TK85,CP200, RINGO e AS1000.

Nos micros acima,destacam-se as seguintes diferenças:

. Os NEZ8000 e alguns TK82C não possuem SLOW.

. O TK85 possui 2K de ROM adicionais,com funções para gravação de programas em alta velocidade (High Speed) e de dados.

. Também existe CP200, versão High-Speed, como o TK85, porém com velocidade diferente.

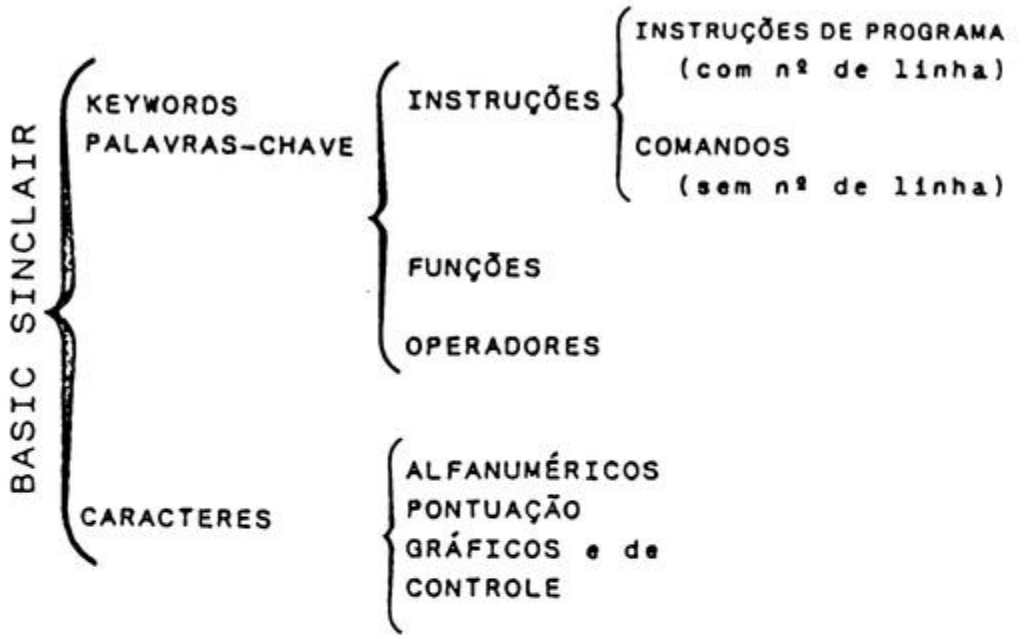
. O RINGO possui teclas multi-função e outros incrementos e grava programas em velocidade superior aos demais, impossibilitando o uso de suas gravações em outros.

No início citei estas máquinas como reproduções do ZX80-8KROM e não do ZX81, pois o ZX81 é uma versão integrada em mais larga escala do ZX80.Utiliza um único chip(circuito integrado),produzido exclusivamente por projeto e para a Sinclair Research,em substituição a mais de 20 circuitos integrados existentes no ZX80 e suas versões nacionais.

Obs.: Alguns TKs foram produzidos com este chip,o famoso SCL - Sinclair Computer Logic.

O Basic Sinclair

O Basic Sinclair divide-se em:



KEYWORDS:

Palavras-chave são as instruções, funções e alguns operadores que usam mais de um caractere, como , por exemplo, AND, OR, NOT e são acionados por uma só tecla.

As Instruções

OS COMANDOS (commands)

Tendo-se um programa, para rodá-lo é preciso ordenar ao sistema , para que o faça, i.e. , para que RODE. Esta "ordem, ou"instrução" dada ao computador, em termos de linguagem de programação, é um COMANDO,no caso, o comando RUN.

COMANDO é uma instrução sem número de linha.

INSTRUÇÕES (statements)

São as linhas do programa. Uma instrução do programa compõe-se de número de linha, instrução e elementos variáveis.

O termo "instrução" causa uma certa confusão, uma vez que no original inglês temos STATEMENT , como linha de programa e INSTRUCTION e ambos os termos tem sido traduzidos como INSTRUÇÃO.

No basic Sinclair todas as instruções podem ser usadas sem número de linha , como comando direto, excluindo-se INPUT. Contudo, algumas instruções não admitem número de linha, sendo apenas utilizáveis como COMANDO DIRETO, como por exemplo, EDIT DELETE, BREAK, etc. ...

AS INSTRUÇÕES DIVIDEM-SE EM:

DE EDIÇÃO (COMANDOS)

EDIT, ↑ , ↓ , →, ←, DELETE OU RUBOUT, NEW LINE OU ENTER

COMANDOS DO SISTEMA

LOAD, SAVE, RUN, CLEAR, NEW, STOP, CONT, FAST, SLOW , NEW LINE LIST, LLIST

DE INPUT/OUTPUT

INPUT, INKEY$, PRINT, PRINT AT, PRINT TAB N, SCROLL, CLS, PLOT N,N, UNPLOT, LPRINT, TAB N;, COPY

DE CONTROLE

PAUSE N, DIM, RAND, GOTO, GOSUB, RETURN, IF ... THEN, TO, NEXT, LET, STEP, FOR

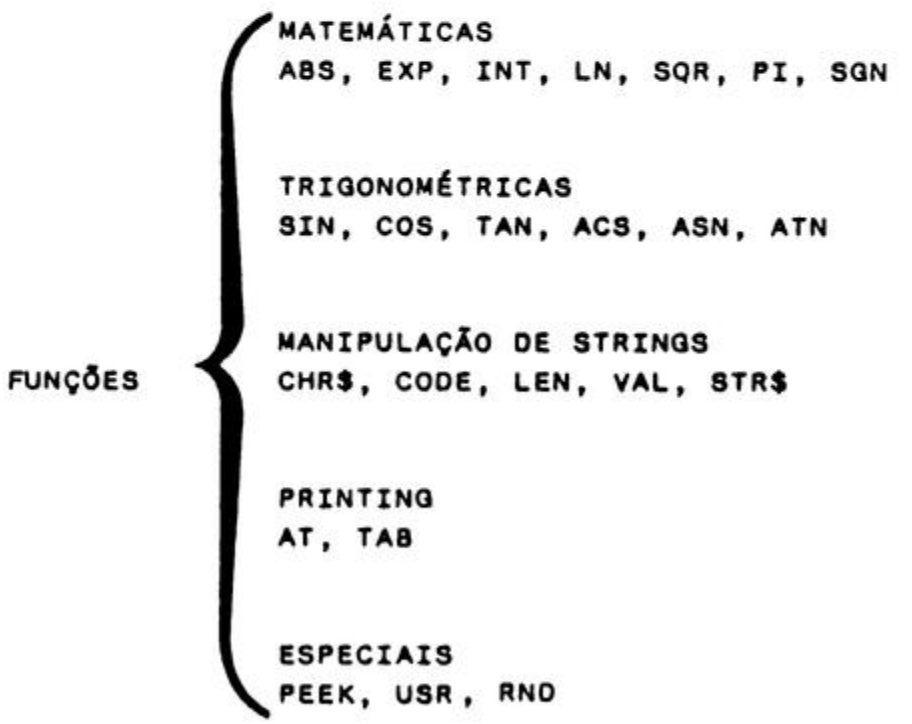
AUXILIARES

POKE, PEEK, REM, USR

As Funções

FUNÇÃO é uma relação matemática entre duas variáveis X e Y, tal que, para cada valor de X corresponda um único valor de Y. Ou, em outras palavras, uma função fornece sempre um certo resultado, para um certo argumento. O argumento de uma função pode ser um número, uma variável, ou uma expressão. No caso de expressão, deve vir entre parênteses.

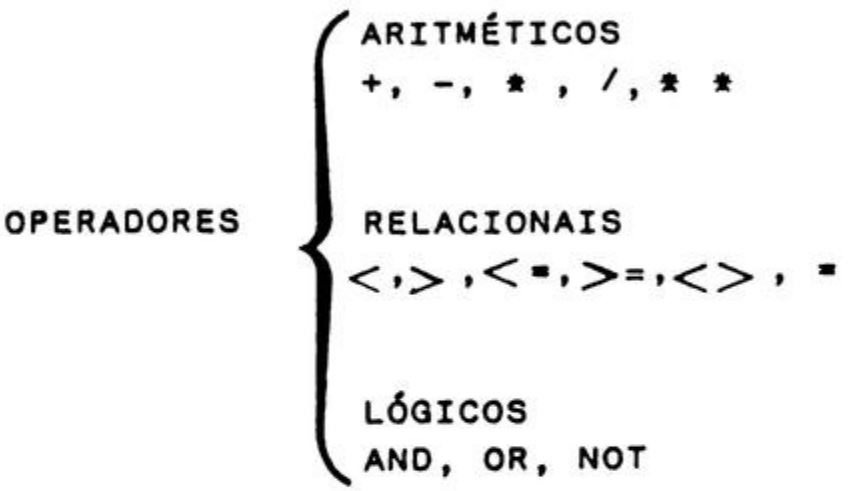
DIVISÃO DAS FUNÇÕES DO BASIC SINCLAIR:



Os Operadores

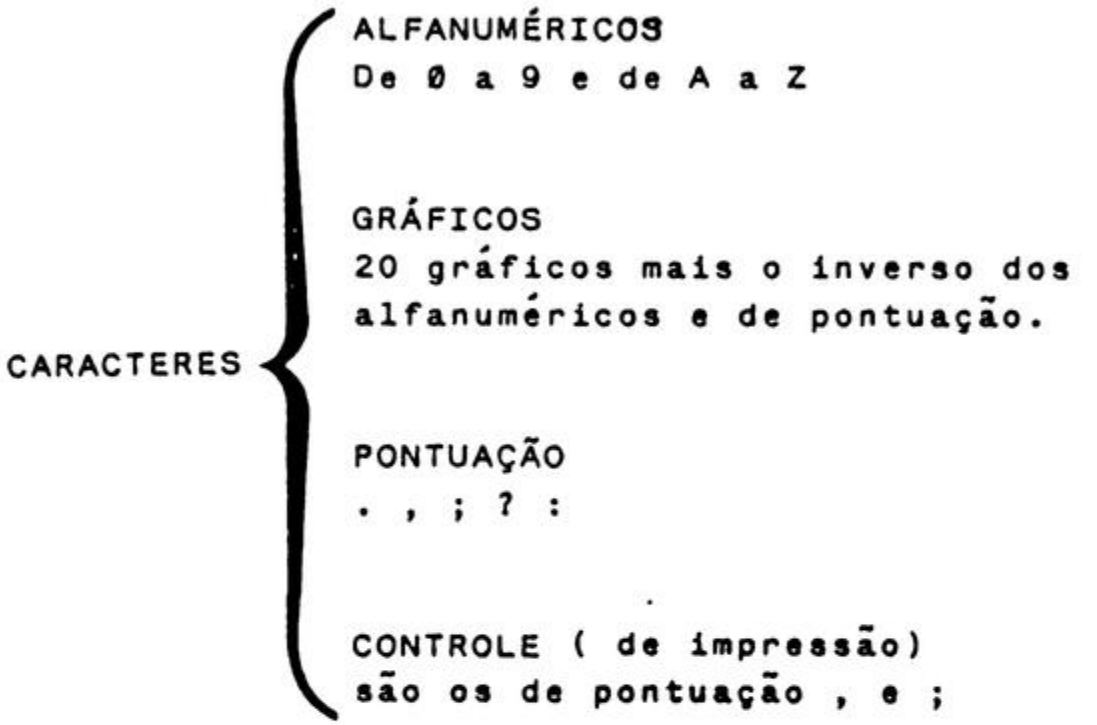
OPERADORES executam as operações aritméticas, relacionais e lógicas entre números , expressões ou strings.

Os operadores dividem-se em:



Os Caracteres

Os caracteres dividem-se em:



KEYWORDS

ABS

Absolute value. É uma função. Fornece o valor absoluto ou módulo de um número. Por exemplo:

```
PRINT ABS-2  resulta em 2,e
PRINT ABS+2  também resulta em 2.
```

Verifique com o programa:

```
10 PRINT "X","ABS X"
20 FOR X=-3 TO 3
30 PRINT X, ABS X
40 NEXT X
```

Que resultará:

```
 X    ABS
-3     3
-2     2
-1     1
 0     0
 1     1
 2     2
 3     3
```

ACS

ARCCOS é o mesmo que ACS. É uma função. Fornece o ângulo em radianos, a partir do cosseno

Programa para plotar a função ARCCOS :

```
10 LET A=-1
20 FOR N=3 TO 42
30 PLOT N, INT(ARCCOS(A)*10)
40 LET A=A+0.05
50 NEXT N
```

Observação:

180 graus = PI radianos

AND

AND é um operador lógico. Forma uma comparação lógica entre números, expressões ou strings. Permite ainda combinar instruções condicionais oferecendo possibilidades de decisões mais complexas.

O valor lógico de uma expressão é interpretado somente com duas possibilidades:

VERDADE = 1 = expressão de valor ≠ Ø
FALSO = Ø = expressão de valor = Ø

Combinando-se duas expressões denominadas A e B por lógica AND, teremos os seguintes resultados, em função das condições de A e B.

A	B	A AND B
F	F	F
F	V	F
V	F	F
V	V	V

O operador lógico AND pode ser usado de diferentes formas. São elas:

Exemplo A:

```
10 IF X=1 AND Y=Ø.5 THEN PRINT"UM E MEI
   O"
```

Obriga que duas condições sejam satisfeitas, simultaneamente, X e Y.

Exemplo B:

```
10 LET X=1AND Y
```

É o mesmo que:

```
10 LET X=1
20 IF Y=0 THEN LET X=0
```

A sentença matemática (1 AND Y) significa que,se Y for 0, a mesma resulta em 0. Caso contrário,assume o valor do primeiro operando, neste caso o valor 1 .
Isto ocorre,porque o operador AND trata o segundo operando,como se fosse um valor lógico do tipo 0 ou 1, respectivamente,verdadeiro ou falso. Se for 0 é falso. Se não for 0, é verdadeiro.

Exemplo C:

```
10 LET X$=Y$ AND A
```

Como no exemplo anterior, se o segundo operando for 0, a expressão assume o valor 0, ou o mesmo que X$="" . Caso o segundo operando não seja 0, X$ será igual a Y$.

Exemplo D:

```
90 LET Y=Y+( INKEY$="M")-(INKEY$="N")
```

Equivale a:

```
90 IF INKEY$="M" THEN LET Y=Y+1
95 IF INKEY$="N" THEN LET Y=Y-1
```

Durante a execução da linha de programa acima, se as teclas M e N não forem pressionadas, as sentenças matemáticas entre parênteses tornam-se falsas e assumem o valor Ø, ficando Y inalterado.
Caso uma das teclas, M ou N, seja pressionada, uma das sentenças será verdadeira e o valor de Y será Y=Y+1 ou Y=Y-1, conforme o caso.

Observação:

Os operadores lógicos do Basic Sinclair tratam somente o segundo operando, como um valor lógico.

ASN

ASN é o mesmo que ARCSIN. É uma função. Fornece o ângulo em radianos, a partir do seno.

Programa para plotar a função ARCSIN

```
10 LET A=-1
20 FOR N=3 TO 42
30 PLOT N, INT(ARCSIN(A)*10)
40 LET A=A+Ø.Ø5
50 NEXT N
```

Observação :

18Ø graus = PI radianos

AT

AT é uma função de impressão. Deve ser usada com a instrução PRINT ou LPRINT. Serve para deslocar a posição de impressão para qualquer linha ou coluna da tela, indo de PRINT AT Ø,Ø ; a PRINT AT 21,31;

Pode ser usado nas formas:

a. Diretamente a seguir de PRINT:

PRINT AT X,Y; "XYZKL" ou

b. Precedida de outros PRINT e AT:

PRINT "KGB"; AT 1Ø,Ø ; "CIA"; AT 2Ø,Ø ; "SNI"

Exemplo:

```
PRINT AT 21,3Ø ;"OK"
```

Irá imprimir OK, na vigésima segunda linha e na trigésima primeira coluna.

Obs.: LPRINT AT interpreta apenas a coluna , uma vez que a impressora não retrocede o papel.

Veja PRINT, LPRINT e TAB.

ATN

ATN é o mesmo que ARCTAN. É uma função. Fornece o ângulo em radianos, a partir da TAN.

Programa para plotar a função ARCTAN.

```
10 LET A=-2
20 FOR N=4 TO 50
30 PLOT N, INT(ARCTAN(A)*10+20)
40 LET A=A+Ø.1
50 NEXT N
```

Observação :

180 graus = PI radianos

BREAK

Break significa interromper. É uma instrução e só pode ser usada como comando direto (sem número de linha).

A tecla de BREAK é a mesma de SPACE.Quando o micro está em RUN,ela funciona como BREAK, interrompendo a execução com o código de reportagem D/Ø . No modo EDIT, ou durante as instruções INPUT, funciona como SPACE.

Para interromper a execução do programa durante as instruções INPUT, usa-se STOP. Durante INPUT (string), o cursor estará na forma"K". É preciso que se apague uma das aspas para que o comando STOP surta efeito.

Observação:

Comando é uma instrução a qual não faz parte de um programa.

Veja o capítulo"Instruções e Comandos".

CHR$

É uma função. Fornece o caractere ou palavra chave de código N. A codificação empregada é própria dos sistemas Sinclair.

Programa para listar os caracteres e palavras chave:

```
10 SLOW
20 FOR N=0 TO 255
30 PRINT AT 20,0;N,CHR$ N
40 SCROLL
50 NEXT N
```

Observação :

O código do sinal de interrogação (?) é 15. Os demais códigos que se apresentarem com uma interrogação não são usados ou representam palavras chave não imprimíveis, como EDIT ou New Line.

CLEAR

Clear é uma instrução. Apaga todas as variáveis. Não significa igualá-las a Ø, mas que ficam indefinidas.

Experimente como comando direto:

```
LET X$="TESTE"
PRINT X$
CLEAR
PRINT X$
```

CLEAR não afeta o programa, apenas as variáveis.

Veja RUN.

CLS

É uma instrução. Limpa a memória de vídeo, (Display File). Não afeta o programa e as variáveis.

Experimente:

```
10 SLOW
20 FOR N=1 TO 20
30 PRINT AT 20,0;"SUMINDO"
40 SCROLL
50 NEXT N
60 CLS
```

Quando a memória em uso estiver quase no limite, as últimas linhas da listagem não serão mais apresentadas no vídeo e pode ocorrer que a linha em edição não seja aceita por New Line. Apague-a e,a seguir, digite CLS. Sem nenhuma listagem no vídeo, a nova linha será "editada".

A razão disto é que,após a aplicação de CLS, surgiram alguns bytes de memória disponível, provenientes do D.File.

Veja"A Memória de Vídeo",na instrução PRINT.

CODE

É uma função para a manipulação de string. O argumento deve ser uma string. CODE fornece o código número de um caractere, palavra chave, ou de um certo caractere de uma string.

Pode ser usado nas formas:

CODE X :fornece o código do caractere X, ou do solicitado.

CODE "ABC" :fornece o código do primeiro caractere da string "ABC".

CODE X$:fornece o código do primeiro ou do único caractere da string X$.

CODE X$(5):fornece o código do quinto caractere da string X$, ou o do solicitado.

CODE irá sempre retornar um número entre Ø e 255, excluindo-se os casos de erro.

Veja CHR$.

CONT

É uma instrução. CONT faz o programa continuar, após ter sido parado por STOP.

Se o programa parou por erro, sendo o código de reportagem ≠ de 9/ ou D/, ele não continuará por CONT, ocorrendo novo erro.

COPY

É uma instrução. Copia exatamente o que estiver no vídeo, para a impressora. É a instrução de impressão mais utilizada, permitindo inclusive a cópia de gráficos, resultados mostrados no vídeo ou mesmo das listagens.

Quando a instrução COPY é acionada, desde que exista impressora acoplada ao micro, a mesma reproduzirá uma tela completa de vinte e duas linhas com trinta e dois caracteres cada, ainda que estes sejam espaços. Caso não exista impressora acoplada, podem ocorrer duas coisas:

1. Nada, continuando o programa na instrução seguinte, ou
2. A "tela ficar branca". Neste caso digite BREAK para retornar.

Veja LPRINT e LLIST.

COS

É uma função. Fornece o cosseno do ângulo dado, em radianos.
A seguir,programa para plotar a função cosseno:

```
10 FOR N=0 TO 6 STEP 0.1
20 PLOT N*10+2,INT(COS(N)*10+20)
30 NEXT N
```

Observação:

180 graus = PI radianos.

DELETE

É o mesmo que RUBOUT. É um comando de edição. Não pode ser usado como linha de programa. Com o micro no modo de edição, permite apagar o caractere à esquerda do cursor. DELETE encontra-se na tecla SHIFT Ø.

Para se apagar uma linha inteira de programa,não é necessário apagar cada um de seus caracteres. Digite apenas o número da linha e, a seguir, New Line (ENTER).

DIM

Dimension. É uma instrução. Reserva espaço na memória para o uso de variáveis indexadas (matrizes). É usada na inicialização. Apaga todos os valores das variáveis, se anteriormente dimensionadas.

O basic Sinclair possui matrizes multidimensionais:

. numéricas - de A(N) a Z(N) - subscritos iniciando-se em 1.

. alfanuméricas - de A$(N) a Z$(N) - subscritos iniciando-se em 1.

Por exemplo: DIM A(1Ø) inicializa uma matriz de nome A com 1Ø elementos. Ou seja, existem 1Ø variáveis subscritas a A, denominadas A(1), A(2), A(3), A(4) ... A(1Ø).

Exemplos de programa com uma Matriz para 1Ø valores:

```
DIM A(1Ø)
FOR N=1 TO 1Ø
PRINT N
INPUT A(N)
PRINT A(N)
NEXT N
```

Exemplo de Programa com uma matriz numérica bidimensional para 12 valores:

```
DIM A(3,4)
FOR N=1 TO 3
FOR X=1 TO 4
INPUT A (N,X)
PRINT A (N,X)
NEXT X
NEXT N
```

DIM A(3,4) dimensiona as 12 variáveis:

A (1,1) A(1,2) A(1,3) A(1,4)
A (2,1) A(2,2) A(2,3) A(2,4)
A (3,1) A(3,2) A(3,3) A(3,4)

Para matriz alfanumérica, proceda como para as variáveis alfanuméricas. Apenas use o caractere $ a seguir da letra de designação da mesma.

CONSUMO DE MEMÓRIA NA ÁREA DAS VARIÁVEIS

Matrizes numéricas custam 4 bytes mais duas vezes a quantidade de dimensões , mais cinco vezes o número de elementos.Exemplos:

A(100) custa 4 + 2*1 + 5*100 = 506 bytes
A(100,7) custa 4 + 2*2 + 5*100*7=3508 bytes
A(100,5,7)custa 4+2*3+5*100*5*7=1751ϕ bytes.

Matrizes alfanuméricas custam 4 bytes mais duas vezes o número de dimensões mais a quantidade de caracteres. Exemplos:

A$ (100) custa 4 + 2*1 + 100 = 106 bytes.
A$ (5,100) custa 4+2*2+5*100 = 508 bytes.
A$ (100,5,7)custa 4+2*3+100*5*7=3510 bytes.

EDIT

Um dos pontos altos do Basic Sinclair, são as facilidades de edição, em muito superiores a outros micros de maior porte.

A tela do micro está composta de 24 linhas de 32 caracteres cada, ficando as duas inferiores reservadas para edição.

As facilidades de edição são:

A. Para alterar a linha em edição.
   Pode-se alterar a posição do cursor por SHIFT 5 ou SHIFT 8. Para inserir caracteres ou palavras chave, apenas digite o desejado e será inserido à esquerda do cursor. Para apagar à esquerda do cursor use SHIFT Ø (DELETE).

B. Para alterar uma linha da listagem.
   Desloque o cursor existente entre o número da linha e as palavras chave até a linha desejada por SHIFT 7 ou SHIFT 6 , ou use LIST N, onde N é o número da linha desejada. Para colocá-la no modo de edição, use EDIT por SHIFT 1 .

O micro sempre estará no modo de EDIÇÃO, exceto durante RUN.

Veja RUBOUT/DELETE, New Line , LIST e CLS .

EXP

Exponential é uma função. Calcula e^{x}, onde X é o argumento dado. A função EXP é o inverso de LN. EXP sempre retorna um número positivo.

```
10 PRINT EXP (3.3) é igual ou semelhante a:
10 PRINT (2.7182818 * * 3.3)
```

Programa para plotar a função EXP:

```
10 LET A=-4
20 FOR N=2 TO 62
30 PLOT N, INT(EXP(A)*5+2)
40 LET A=A+Ø.1
50 NEXT N
```

Observação:

e = 2.7182818

FAST

Os microcomputadores,baseados no sistema operacional do ZX81, possuem duas velocidades de operação: FAST e SLOW.

Quando o micro é ligado, encontra-se no modo do SLOW. Isto é, o Display File é constantemente enviado à tela de TV. O comando FAST muda para o modo FAST. Neste modo, o processamento é que é feito constantemente , sendo o D.file enviado à TV somente durante as instruções INPUT, PAUSE ou , quando não houver alguma coisa a ser executada.Isto significa que, durante o"processamento" a tela ficará "em branco".

No modo FAST a execução é cerca de quatro vezes mais rápida que no modo SLOW.

Observação:

Os microcomputadores NEZ8000 e alguns TK82C só operam no modo FAST, por diferenças de hardware, apesar de possuirem o mesmo sistema operacional.

FAST e SLOW não alteram a velocidade de processamento (da CPU).

Veja SLOW.

FOR

É uma instrução usada na definição de "loops" e requer o uso em conjunto de TO e NEXT.

Exemplo de LOOPING:

```
10 FOR N=1 TO 10
20 PRINT N
30 NEXT N
```

A variável N, utilizada no exemplo, é a chamada variável de controle.

Quando a linha 10 for executada, N assumirá o valor inicial UM e, ao atingir a instrução NEXT N, será incrementada em 1. Se o valor atingido for menor que o valor especificado após o TO, o programa volta à linha seguinte à instrução FOR ... TO.Caso contrário, continua após o NEXT.

O exemplo acima é o mesmo que:

```
10 LET N=1
20 PRINT N
30 LET N=N + 1
40 IF N<11 THEN GOTO 20
```

Veja STEP.

FUNCTION

É um comando do sistema. Comuta o teclado para o modo F , por SHIFT FUNCTION. Uma vez acessada uma única função, na forma KEY WORD, o cursor volta ao modo K .

As funções do cursor F , modo FUNCTION , são todas as palavras-chaves estampadas logo abaixo de cada uma das teclas.

Veja as Funções do Basic Sinclair, no início deste volume.

Observação:

No CP 200 as funções estão acima das teclas.

GOSUB

Instrução de desvio de programa, para uma sub-rotina indicada, a ser terminada por RETURN. Por exemplo:

```
 10 INPUT X$
 20 GOSUB 100
 30 STOP
 40 PRINT X$
100 PRINT X$
110 RETURN
```

O número de linha de destino, indicado a seguir de GOSUB, pode ser indicado por uma variável ou sentença matemática. Caso a linha de número indicado não exista, o programa continua no número de linha subsequente à indicada.

Veja GOTO.

GOTO

Inicia, reinicia ou desvia a execução do programa para a linha de número especificado a seguir da mesma, sem alterar as variáveis.

O número de linha de destino pode ser indicado por uma variável ou sentença matemática. Caso a linha de número indicado não exista, o programa continua na linha subsequente à indicada.

Exemplos de GOTO:

```
GOTO 100
GOTO N
GOTO 100 * N
```

A instrução GOTO pode ser condicionada pelo uso de IF ... THEN GOTO ...
Por exemplo: IF IDADE > 8 THEN GOTO 900

Substituindo ON ... GOTO inexistente no Basic Sinclair:

```
10 PRINT "OPÇÕES DO MENU"
20 PRINT"DIGITE DE 1 A 5"
30 INPUT A
40 GOTO A * 100 + 1000
```

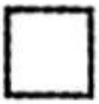
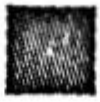
As respostas do menu, de 1 a 5, desviarão o programa para as linhas 1100, 1200,1300, 1400 e 1500 respectivamente.

GRAPHICS

É um dos modos do teclado/cursor. Digitando-se SHIFT 9, o cursor muda para G, ficando disponíveis os caracteres gráficos.

Ou seja, o inverso de todos os caracteres e símbolos (excluindo Keywords), acessados diretamente e vinte símbolos gráficos com acesso por SHIFT, tudo com o cursor em G.

Os símbolos gráficos estão ilustrados a seguir, com seus códigos e suas teclas de localização.

SÍMBOLO	CÓDIGO	MODO	SÍMBOLO	CODIGO	MODO
	0	G ou L Espaço		128	G Espaço
	1	G Shifted 1		129	G Shifted Q
	2	G Shifted 2		130	G Shifted W
	3	G Shifted 7		131	G Shifted 6
	4	G Shifted 4		132	G Shifted R
	5	G Shifted 5		133	G Shifted 8
	6	G Shifted T		134	G Shifted Y
	7	G Shifted E		135	G Shifted 3
	8	G Shifted A		136	G Shifted H
	9	G Shifted D		137	G Shifted G
	10	G Shifted S		138	G Shifted F

IF

É uma instrução que permite ao programador executar uma tomada de decisão. É uma das mais importantes do Basic. Deve ser usada na forma:

IF CONDIÇÃO THEN INSTRUÇÃO, traduzindo-se:

SE a condição for satisfeita, ENTÃO execute a instrução determinada.

A instrução a seguir do THEN pode ser qualquer uma, inclusive outro IF.

Exemplo:

```
 10 PRINT "SUA IDADE ?"
 20 INPUT X
 30 CLS
 40 IF X >=18 THEN GOTO 100
 50 PRINT "VOCE EH MENOR"
 60 GOTO 10
100 PRINT "VOCE EH MAIOR"
110 GOTO 10
```

Veja um exemplo de FOR...NEXT com IF...THEN.

INKEY$

INKEY$ é uma função. Lê o valor do teclado, ou melhor, da tecla digitada, quando é executada . Quando nenhuma tecla for pressionada, retorna STRING NULA, = "".

Com tecla pressionada ou não, o microcomputador continua a execução do programa.

Caso se deseje aguardar que uma tecla seja pressionada, deve-se colocar o INKEY$ no interior de um looping. Por exemplo:

```
 5 LET T=1
10 SLOW
20 PRINT "DIGITE UMA TECLA PARA STOP"
30 LET X$ = INKEY$
40 IF X$ = "" THEN GOTO 80
50 PRINT X$
60 LET T = T+1
70 PRINT AT 20,0; T
80 GOTO 20
```

Para se manter algo impresso no vídeo, durante o looping com INKEY$, é preciso que o microcomputador esteja no modo SLOW.

Os microcomputadores TK80, NEZ80, NEZ8000 e os primeiros TK82C não possuem SLOW e para o uso de INKEY$ existe o seguinte macete:

```
10 FAST
20 PRINT "DIGITE UMA TECLA"
30 RAND USR 681
40 LET X$ = INKEY$
50 PRINT X$
```

O uso de USR 681 pelas linhas 30 e 40 significa o mesmo que INPUT INKEY$. É INKEY$ "parado".

Vide SLOW e FAST.

INPUT

A instrução INPUT interrompe a execução do programa para uma entrada de dados via teclado. A entrada de dados é sempre terminada por NEW LINE/ENTER.

Durante as instruções INPUT, mesmo no modo FAST, o DFILE é apresentado no vídeo, porque a execução do programa foi parada.

Os dados a serem entrados podem ser valores numéricos ou alfanuméricos, sendo colocados como variáveis do tipo X, X$ ou subscritas.

Podemos dizer que, de um certo modo, a instrução INPUT define variáveis pela entrada de dados do teclado.

Possibilidades de INPUT :

A. INPUT X

INPUT X ou INPUT variável numérica de qualquer nome permitido, entra um valor numérico qualquer entre $\pm 3 \times 10^{39}$ a $\pm 7 \times 10^{38}$, ou o nome de uma variável.

Neste caso, quando o microcomputador parar para o INPUT, será apresentado no vídeo o cursor L.

Se você desejar parar a execução do programa, digite qualquer letra e NEW LINE ou use STOP.

B. INPUT X$

INPUT X$ ou INPUT variável alfanumérica de qualquer nome permitido.

Neste caso, quando o microcomputador parar , para o INPUT, será mostrado o cursor entre aspas. Se você desejar parar a execução do programa, não adiantará usar STOP, a menos que, antes, você apague uma das aspas.

C. INPUT X(X,...,Z) e INPUT X$(Y,...,Z)

Permitem a entrada de variável numérica ou alfanumérica subscrita (ou indexada) de uma ou mais dimensões, mas um único elemento por vez, desde que anteriormente dimensionada.

Experimente:

```
10 LET A=10
20 PRINT "A=";A
30 INPUT A
40 SCROLL
50 GOTO 15
```

Veja DIM.

INT

É uma função. Retorna a parte inteira do número dado como argumento.

Por exemplo:

```
PRINT INT 3.1 resulta em 3 e
PRINT INT 3.6 também resulta em 3 .
```

Para fazer aproximações aritméticas do tipo 3.1≅3 e 3.6≅4, utilize a regra:

```
LET X=INT(X+Ø.5)
```

LEN

Função para manipulação de strings.

O comprimento de uma determinada string pode ser obtido por LEN.

LEN A$ fornece o número de caracteres da string A$.

String nula retorna LEN Ø . Por exemplo:

```
LET A$ = ""    e
PRINT LEN A$   resulta em Ø .
```

LET

A instrução LET define um valor para uma variável. Trabalha com variáveis numéricas e alfanuméricas, indexadas ou não. Também define uma variável, por outra variável já definida.

Nenhuma variável pode ser usada , a menos que anteriormente definida por LET, INPUT, ou por DIM no caso de subscritas.

Possibilidades de uso de LET:

1. LET X = Y

Define X como sendo igual a Y.
Y pode ser um número, variável, ou o resultado de expressão a ser calculada , mesmo que envolva números e variáveis.

Exemplos:

```
LET X = Ø
LET X = A
LET X = A + 1
LET X = X + A
LET X = 2*(X+A)+A
```

X e Y também podem ser variáveis indexadas do tipo X(1), ou multidimensional X(1,1).

2. LET X$ = Y$.

Define X$ como igual a Y$.

Y$ pode ser uma string, uma variável alfa numérica, ou resultado de uma operação com strings.

Exemplos:

```
LET X$ = "TESTE"
LET X$ = Y$
LET X$ = Y$ + "TESTE"
LET X$ = X$ + Y$
```

X$ também pode ser uma variável indexada.

3. LET X$ (N) = Y$

Torna o caractere N da string X$ igual a Y$.

Por exemplo:

```
10 LET X$ = "ABCDF"
20 LET Y$ = "X"
30 LET X$(2) = Y$
40 PRINT X$
```

Resulta em AXCDEF.

4. LET X$ (2 TO 4) = Y$

Neste caso, Y$ deve ter 4 caracteres, os quais assumirão a string X$, do 2º ao 4º caractere .

Por exemplo:

```
10 LET X$ = "ABCDEFG"
20 LET Y$ = "XXX"
30 LET X$ (2 TO 4) = Y$
40 PRINT X$
```

Retorna AXXXEFG.

LIST

LIST é um comando do sistema. Permite listar o programa no vídeo, do início, ou a partir da linha especificada.

LIST lista o programa a partir da primeira linha e pára com o código 5/ ou 4/, se houver um esgotamento do vídeo. Digite CONT e NEW LINE.

LIST n lista o programa, a partir da linha de número indicado ou subsequente, no caso da inexistência desta. LIST n também desloca o cursor para a linha indicada, alterando a variável do sistema "current line".

LLIST

LLIST é uma instrução. Funciona igualmente a LIST, sendo a listagem do programa enviada à impressora, ao invés de ao vídeo.

LLIST lista o programa todo.

LLIST 100 lista a partir a linha 100.

Veja COPY e LPRINT.

LN

É uma função que retorna o logarítimo natural do argumento X.

Observação:

LN (X) = Log e (X) e = 2,7182818

Note que o logarítimo comum:
log 10 (X) = (Ln(X))/(LN 10)

A função LN é o inverso de EXP e,portanto, se:

EXP (X) = Y então
(X) = LN(Y)

LN(Y) é o logarítimo natural de Y. O anti-log é EXP(LN(Y)).

Programa para plotar a função LN:

```
10 FOR X = . 5 TO 30 STEP.5
20 PLOT X * 2, INT(LN X * 10)
30 NEXT X
```

LOAD

O comando LOAD é usado para carregar no microcomputador os programas previamente gravados em cassete.

Usado na forma:

LOAD ""
carrega o primeiro programa do cassete.

Na forma:
LOAD "NOME XY"
carrega o programa de nome indicado.

Procedimentos:

Ligue primeiramente o gravador,depois acione NEW LINE ou ENTER.

Após carregar o programa, na maioria dos casos, o microcomputador pára com o código de reportagem Ø/Ø.

Caso a gravação do programa tenha sido feita no modo RUN, o programa sairá rodando. Para carregar gravações deste tipo e fazê-las parar imediatamente, após a carga,use: FAST e RAND USR 836.

No vídeo serão reproduzidos dois tipos de imagem : um , durante o vazio da fita , entre programas e outro do sinal do programa.

É importante saber distinguir, no vídeo, a imagem de uma boa gravação.

LPRINT

A instrução LPRINT é semelhante a PRINT, só que a impressão é executada na impressora, ao invés de no vídeo.

Assim como PRINT permite o uso de vírgula, ponto e vírgula e TAB, LPRINT AT interpreta apenas o número da coluna, sendo ignorado o número de linha, porque a impressora nao pode retroceder o papel.

O comando BREAK interrompe a execução de LPRINT.

Se executar LPRINT sem a impressora e a tela ficar branca, também use BREAK.

As instruções LPRINT não são executadas diretamente. Um Buffer existente na RAM, nos endereços 16444 a 16472, armazena 32 caracteres que serão enviados à impressora, somente em um dos casos:

1. Quando o Buffer estiver completo, com uma linha de 32 caracteres.
2. Após as instruções LPRINT, não terminadas com vírgula, ou ponto e vírgula.
3. Quando a função TAB, ou caractere de controle vírgula, exigir uma nova linha.
4. Ao término do programa, se ficou alguma coisa a ser impressa, mesmo que não se

ja uma linha completa.

Programa exemplo da existência do Buffer:

```
10 LPRINT "TESTE";
15 LPRINT "DO BUFFER";
20 PRINT "O PROGRAMA CONTINUOU E A
   LINHA NÃO FOI IMPRESSA"
25 PAUSE 500
30 PRINT "O PROGRAMA ACABOU, E O BUFFER
   ", "EH ENVIADO A IMPRESSORA"
```

ERROS DE LPRINT:

com números fracionários,menores que 0,01

Teste:

```
10 LET X = 0.0001
20 LPRINT X
```

resulta em .0XY1

Para corrigir use:

```
10 LET X = 0.0001
20 LET X$ = STR$ X
30 LPRINT X$
```

Vide COPY e LLIST.

NEW

O comando NEW apaga o programa, as variáveis e limpa a tela.

NEW apaga a memória dos endereços de 16509 até o dado pela variável do sistema operacional, denominada RAMTOP.

NEW LINE

É um comando do sistema . Deve ser usado:

1. Ao final de um comando direto.

2. Ao final de uma instrução de programa.

3. Ao final de uma entrada de dados.

Em todos os casos, após New Line, o micro verifica a validade da instrução ou dado entrado, executando-o ou retornando com o código de erro.

NOT

NOT é um operador lógico. É uma negação lógica de números, expressões e strings.

O valor lógico de uma expressão é interpretado somente com duas possibilidades:

```
VERDADE =1= Expressão de valor ≠ Ø
FALSO   =Ø= Expressão de valor = Ø
```

Exemplos de uso de NOT:

1. Como operador numérico:

```
10 LET A = NOT B
```

Se B não for Ø, então A assume valor Ø .
Se B for Ø , então A assume valor 1.

É o mesmo que:

```
10 LET A = Ø
15 IF B = Ø THEN LET X = 1
```

NOT trata o operando B, como um valor lógico do tipo Ø ou 1 e então a expressão assume o valor lógico oposto.

2. Com strings (negando a igualdade)

```
10 IF NOT A$=B$ THEN PRINT "DIFERENTE"
```

É o mesmo que:

```
10 IF A$<>B$ THEN PRINT "DIFERENTE"
```

OR

OR é um operador lógico. Combina por lógica "OR" números, expressões ou strings. Permite ainda combinar instruções condicionais, oferecendo possibilidades de decisões mais complexas.

O valor lógico de uma expressão é interpretado somente duas possibilidades:

VERDADE = 1 = expressão de valor ≠ Ø
FALSO = Ø = expressão de valor = Ø

Combinando-se duas expressões, denominadas A e B por lógica OR, teremos os seguintes resultados, em função das condições A e B:

A	B	A OR B
F	F	F
F	V	V
V	F	V
V	V	V

As possibilidades de uso do operador OR são:

1. Operador Numérico:

```
LET X=3 OR Y
```

Significa que, se Y for Ø, então X assumirá

valor 3 . Caso Y seja diferente de Ø,então X assumirá valor 1.

LET X = 3 OR Y é o mesmo que

LET X=1

IF Y = Ø THEN LET X = 3

No Basic Sinclair, somente o segundo operando é tratado como um valor lógico do tipo Ø ou 1.

2. Em expressões e,em especial, combinando condicionais com IF com números e strings.

   IF X$ = "PRETO" OR X$ = "MARRON" THEN PRINT "COR ESCURA"

Veja AND e NOT.

PAUSE N

É uma instrução.Pára o programa por um certo período de tempo especificado. PAUSE N pára o programa por N/60 segundos.

O valor máximo admissível de PAUSE é 32767. ≅ 9 minutos.

Valores maiores equivalem à PAUSE permanente. Durante uma PAUSE, pressionando-se qualquer tecla, o programa continua.

Programa exemplo:

```
10 SLOW
20 PRINT "QUANTOS SEGUNDOS DE PAUSE?"
30 INPUT X
40 PAUSE X * 60
50 PRINT "JAH SE FORAM"; X ;"SEGUNDOS"
```

Teste:

```
10 PRINT "DIGITE UMA TECLA"
20 PAUSE 40000
30 PRINT "FIM".
```

Mesmo no modo SLOW,quando executando PAUSE, o vídeo "dá uma piscada".

PEEK

PEEK é uma função do sistema. Retorna o conteúdo de um especificado endereço da memória.

O valor máximo retornável por PEEK é 255, porque o microcomputador é baseado no Z80, um microprocessador de 8 bits. Assim, toda a memória do sistema foi organizada em bytes de 8 bits. A cada byte corresponde um endereço. Bit significa (Binary Digit), dígito binário. Em 8 bits só é possível representar de 0 a 255 decimal. Valores maiores requerem mais de 1 byte ou endereço, para sua representação.

Usando PEEK, você poderá ler qualquer endereço da memória (ROM e RAM).

Exemplo:

```
10 FOR N=0 TO 8191
20 PRINT AT 20,0; N, PEEK N
30 SCROLL
40 NEXT N
```

Para ler valores armazenados em dois endereços, como os de algumas variáveis do sistema, deve-se multiplicar o conteúdo do endereço mais alto por 256 e somá-lo ao do anterior.

Por exemplo:
Variável RAM TOP, endereços 16388 e 16389, usa-se:

```
PRINT PEEK 16388 + 256 * PEEK 16389.
```

Em dois bytes o valor máximo admissível é 65535.

PI

PI é uma função. Não requer argumento. Retorna o valor 3,141592653. O valor é armazenado em 10 dígitos, mas imprime no vídeo apenas 8.

Para converter graus em radianos, use:

```
10 INPUT GRAUS
20 PRINT GRAUS * PI/180
```

PLOT

É uma instrução.

As posições de PLOT vão de 0,0 a 63,43.

Note que PLOT X,Y significa que X é posição horizontal e Y vertical. O contrário de PRINT AT X,Y onde X é linha (posição vertical) e Y é coluna (posição horizontal).

A seguir, programa para preencher todas as posições de PLOT:

```
05 SLOW
10 FOR N = 0 TO 63
20 FOR X = 0 TO 43
30 PLOT N,X
40 NEXT X
50 NEXT N
```

Altere:

```
10 FOR N = 0 TO 63 STEP 2
20 FOR X = 0 TO 43 STEP 2
```

Veja UNPLOT.

POKE

É uma instrução. Armazena o valor dado no endereço especificado. Por exemplo:

POKE 16515,Ø armazena o valor Ø no endereço 16515.

Existem basicamente dois tipos de memória em seu micro. ROM e RAM. A ROM é inalterável.

Os endereços de RAM iniciam-se em 16384 e vão:

para 2K de RAM de 16384 a 18431
para 16K de RAM, de 16384 a 32767
para 48K de RAM, de 16384 a 65535

Todavia, você não pode aplicar POKE indiscriminadamente, em qualquer endereço, porque na RAM estão armazenados vários ítens, como:

. as variáveis do sistema
. o programa do usuário
. as variáveis do programa usuário
. D.FILE (memória de vídeo)
. Stack do calculador, da máquina , etc...

POKE ENDEREÇO, VALOR

Aceita endereços de Ø a 65535 ou retorna código de erro B/ .

Aceita valores de -255 a +255. Se o número for negativo, o micro irá somá-lo a 256, an-

tes de armazená-lo. Números fora desta faixa retornam código de erro B/ .

Você pode aplicar POKE nos endereços da ROM, mas não terá efeito algum.

Alguns exemplos de POKE:

A.

```
01 REM XXXXXXXXXXX
10 FOR N = 1 TO 10
20 INPUT X$
30 POKE N, CODE X$
40 PRINT X$
50 NEXT N
60 CLS
70 LIST
```

B.

```
05 REM VÍDEO DE 24 LINHAS
10 POKE 16418,0
20 FOR N = 0 TO 23
30 PRINT , N
40 NEXT N
```

C.

```
1 REM TESTE
POKE 16510,0    para transformar a linha
                1 em 0 .
```

PRINT

É uma instrução. Envia ao vídeo um caractere ou conjunto de caracteres. Oferece as seguintes possibilidades:

a. PRINT: Imprime uma linha em branco.

b. PRINT N ou N $: Imprime variáveis e constantes numéricas.

c. PRINT "XYZ": Imprime cadeia de caracteres.

Para cada linha de PRINT terminada por NEW LINE ou ENTER, como nos modelos acima, a posição de impressão é sempre deslocada para a linha seguinte. Por exemplo:

```
10 PRINT "TESTE"
20 PRINT "TESTADO"
30 PRINT
40 PRINT "FIM"
```

resultará em:

```
TESTE
TESTADO

FIM
```

Em conjunto com a instrução PRINT, usam-se os caracteres de controle (de impressão) , (vírgula) e ; (ponto e vírgula), com o ob

jetivo de se continuar a impressão sem mudar de linha.

As instruções PRINT, quando terminadas por ; (ponto e vírgula), fazem com que a próxima impressão seja a seguir da última.
Exemplo:

```
10 PRINT "TESTE " ;
20 PRINT "TESTADO"
```

resulta em:

```
TESTE TESTADO
```

Experimente:

```
10 FOR N = 1 TO 20
20 PRINT CHR$ N
30 NEXT N
```

e altere:

```
20 PRINT CHR$ N;
```

A vírgula desloca a posição de impressão 16 posições à frente. Teste:

```
10 FOR N = 1 TO 20
20 PRINT N,
30 NEXT N
```

Veja o efeito de:

```
10 PRINT "TESTE","TESTADO"
```

o mesmo que:

```
10 PRINT "TESTE",
20 PRINT "TESTADO"
```

Ainda em auxílio das facilidades de impressão , o Basic Sinclair oferece as funções (de impressão) AT e TAB, respectivamente,

PRINT AT Y,X ; e PRINT TAB X;

onde X é coluna e Y é linha.

PRINT AT possibilita alterar a posição de impressão (PRINT POSITION) para qualquer linha e coluna da tela e PRINT TAB permite deslocar a impressão X colunas, se exceder a linha, desloca para a seguinte.

Veja AT e TAB.

A MEMÓRIA DE VÍDEO (D.FILE)

Tudo que é impresso no vídeo, ou melhor, o vídeo como você o vê, com 24 linhas de 32 caracteres correspondem a 24 x 33 bytes na memória de vídeo. Uma linha ocupa 33 bytes, sendo 32 para os caracteres e 1 para NEW LINE.

Cada caractere da tela ocupa um determinado byte na memória. Os endereços da memória de vídeo variam de tempos em tempos, por questões de FIRM e HARDWARE. O endereço de

início, ou do primeiro byte da memória de vídeo (D.File) é dado pela variável do sistema D.File, endereços 16396 e 16397.

O D.File começa com o primeiro byte em valor 118 (NEW LINE) e também termina por 118.

O programa, a seguir, lê os 32 primeiros bytes da memória de vídeo e inverte os caracteres, acrescendo 128 aos seus códigos.

Rode em SLOW;

```
10 LET DFILE = PEEK 16396 + 256 * PEEK
   16397
20 PRINT "QUALQUER COISA"
30 REM INVERTER UMA LINHA DO VÍDEO
40 FOR N=1 TO 32
50 LET X=PEEK (DFILE + N)
60 IF X<>118 THEN POKE DFILE+N,X+128
70 NEXT N
```

A posição de impressão, ou endereço da memória de vídeo onde a próxima instrução PRINT será aplicada, pode ser identificada pela variável do sistema, denominada PRINT POSITION ou CC,corrent caracter, endereço 16398.

No programa"Fórmula Micro" do livro "30 JOGOS PARA TK82 e CP200", de minha autoria,a mesma é utilizada, para saber se a posição ondè será"impresso o carro" é um vazio(va-

lor Ø), ou se já contém alguma coisa, como a murada da pista, indicando , então, a colisão. PRINT AT e TAB funcionam pela alteração desta variável.

As duas linhas inferiores do vídeo são reservadas para edição, mas podem ser usadas, desde que se altere a variável do sistema, do endereço 16418, cujo valor é habitualmente 2. Esta alteração, por POKE 16418,Ø, por exemplo, não funciona como comando direto, somente como linha de programa. Experimente :

```
1Ø POKE 16418,Ø
2Ø FOR N=Ø TO 23
3Ø PRINT AT N,N;"OK"
4Ø NEXT N
```

Observação:

A variável do endereço 16418 deve voltar ao valor 2, antes de qualquer instrução INPUT ou para edição.

RAND

Rand é uma palavra-chave, usada como instrução para controlar a aletoriedade de RND, iniciando a sequência randômica numa posição desconhecida.

O microcomputador tem uma sequência habitual de 65536 números e RAND N faz RND iniciar a leitura da sequência, a partir da posição N.

Teste por:

```
10 RAND 7
20 LET A = 1
30 PRINT RND
40 LET A = A+1
50 IF A<6 THEN GOTO 30
60 GOTO 10
```

Nada randômico.

REM

REM. É uma abreviatura de Remarks, observação. É uma instrução e pode ser seguida de quaisquer caracteres imprimíveis.

Seu único efeito é aparecer na listagem do programa. Na execução do mesmo, não tem efeito algum.

Habitualmente é utilizado para identificar os programas e/ou suas rotinas.

RETURN

É uma instrução de término de sub-rotina.

Somente é executável, se houver um GOSUB correspondente. Caso contrário, o programa pára com o código de erro 7/ .

As instruções GOSUB colocam o número da linha seguinte no "GOSUB STACK" e então "pulam " para a linha especificada ou subsequente . Quando uma instrução RETURN é executada, o programa desvia-se para a última linha, cujo número foi colocado no " GOSUB STACK".

Exemplo de GOSUB

```
 10 PRINT "DIGITE UM N□ FRACIONÁRIO"
 20 INPUT X
 30 CLS
 40 GOSUB 100
 50 RUN
100 LET Y = INT (X)
110 PRINT X, "INT = "; Y
120 PAUSE 300
130 CLS
140 RETURN
```

Veja GOSUB e GOTO.

RND

É uma função. Retorna um número pseudo-aleatório maior ou igual a zero, mas menor que um.

É pseudo, porque, na verdade,este número é retirado de uma sequência definida.

Veja o uso de RAND.

Exemplos para se obter um número pseudo-aleatório :

entre Ø e 99:

```
10 LET X = INT(RND * 1ØØ)
```

entre Ø e 1ØØ:

```
10 LET X = INT(RND * 1Ø1)
```

entre 1 e 1ØØ:

```
10 LET X = INT(RND * 1ØØ) + 1
```

RUN

É uma instrução. Apaga as variáveis e roda o programa do início, ou , a partir da linha indicada ou subsequente.

Exemplos:

RUN, apaga as variáveis e roda o programa do início.

RUN 1ØØ apaga as variáveis e roda o programa, a partir da linha 1ØØ, ou seguinte, caso esta inexista.

Para rodar um programa do início, ou reiniciá-lo, sem apagar as variáveis, use GOTOn. Vide GOTO.

RUN, como comando direto, apaga o vídeo.Como linha de programa, não.

SAVE

A instrução SAVE envia o programa e suas variáveis, para o cassete à razão de 250 baud.

Se uma cópia do programa for feita em cassete, o mesmo poderá ser posteriormente recuperado por LOAD.

SAVE deve ser usado com uma string de argumento. Por exemplo:

SAVE"NOME" ou
SAVE A$

Para fazer com que seu programa "saia rodando", após recuperá-lo do cassete, coloque SAVE, como a primeira linha do mesmo e grave-o por GOTO 1.

Após a gravação, a última letra do nome do programa aparecerá em caractere inverso.

Para parar estas gravações, ao invés de carregar por LOAD"", use FAST e RAND USR 836.

A instrução SAVE envia para o cassete todos os bytes da RAM do usuário, do início até o último, dado pela variável do sistema, denominada RAMTOP. Os programas gravados com a expansão de 16K, só poderão ser carregados com 16K de RAM, mesmo que sejam menores que 1K.

Nestes casos, ao invés de remover a expansão , é melhor alterar a variável RAMTOP por:

```
POKE 16389,72
```

que o micro pensará ter apenas 2K de RAM. Este procedimento pode ser adotado também para encurtar o tempo de gravação e carga.

Cálculo aproximado do tempo de gravação em segundos é igual ao número de bytes do programa x 8/250.

Sequência de SAVE:

1. Cheque a fita no cassete.
2. Cheque o cabo de ligação do Micro-Mic a ao MIC do cassete.
3. Ajuste agudos para o máximo e volume para 1/4 a 2/4 do máximo.
4. Verifique o programa e digite SAVE "NOME".
5. Digite NEW LINE ou ENTER .
6. Durante os 5 segundos iniciais a tela ficará branca. Neste meio tempo, acione o cassete para REC.

Acionando primeiro o micro e depois o cassete, evita-se a introdução de ruídos no início da gravação. A imagem produzida no vídeo é proposital e serve para monitorar a gravação.

SCROLL

É uma instrução. Desloca todo o Display File uma linha acima. A primeira linha é perdida e a última linha entra vazia.

A posição de impressão é deslocada para o início da última linha.

Teste:

```
10 INPUT X$
20 PRINT X$
30 SCROLL
40 GOTO 10
```

Obs : Não funciona com LPRINT.

SGN

SIGNAL é uma função. Retorna 1, se o argumento for positivo; -1, se for negativo e Ø , se ele for Ø.

Programa exemplo:

```
05 PRINT AT 2Ø; Ø; "N" , "SGN"
10 FOR N=-3 TO 3
15 SCROLL
20 PRINT AT 2Ø; Ø; N , SGN N
25 NEXT N
```

Retorna:

N	SGN
-3	-1
-2	-1
-1	-1
Ø	Ø
1	1
2	1
3	1

SIN

É uma função. Fornece o seno do ângulo dado, em radianos.

A seguir, programa para plotar a função seno:

```
10 FOR N = 0 TO 6 STEP 0.1
20 PLOT N * 10 + 2, INT(SIN(N)*10 + 20
30 NEXT N
```

SLOW

Os microcomputadores baseados no sistema operacional do ZX81 operam em duas velocidades. FAST e SLOW.

Quando o micro é ligado, ele se encontra no modo SLOW. É o modo em que o vídeo se apresenta continuamente, sem piscar e o processamento é feito entre cada imagem vídeo, ficando 4 vezes mais lento que no modo FAST.

Já no modo FAST, o processamento é feito ininterruptamente e o vídeo é apresentado somente quando este termina, ou durante comandos de INPUT e PAUSE.

Observação:

Os microcomputadores NEZ8000 e alguns TK82 C não possuem o modo SLOW,por diferenças de circuito,apesar de terem o mesmo sistema operacional. Ou melhor, por serem sucessores do ZX80 com a 8K ROM do ZX81 e não do ZX81.

SQR

SQR é uma função. Significa Square Root (raiz quadrada) do número dado (n). O inverso de raiz quadrada é a multiplicação de um número por ele mesmo.

Programa para plotar a SQR de X, com X variando de (Ø a 6Ø) * 5 + 2:

```
1Ø FOR X = Ø TO 6Ø
2Ø PLOT X + 2, (SQR X) * 5 + 2
3Ø NEXT X
```

Fornece somente as respostas positivas e não as negativas. Por exemplo:

$\sqrt{25}$ tem como resposta 5 e -5, porque

5 x 5 = 25 e -5 x -5 = 25

O Basic Sinclair retorna código de erro B para SQR de números menores que Ø (ou negativos).

STEP

STEP é uma instrução para uso em conjunto com FOR ... TO . Veja o uso de FOR...TO... NEXT .

```
10 FOR N = -5 TO +5
15 PRINT N
20 NEXT N
```

A variável de controle N é habitualmente implementada ou decrescida de 1 em 1.Com o uso de STEP é possível alterarmos o valor de acréscimo ou decréscimo da variável de controle. Por exemplo:

```
10 FOR N = -5 TO +5 STEP 0.5
15 PRINT N
20 NEXT N
```

Fará com que a variável de controle seja incrementada de 0,5 em 0,5.

STOP

É uma instrução. Pára a execução do programa com o código de reportagem 9/ . Por CONT o programa continua na próxima linha.

Experimente:

```
 5 CLS
10 PRINT "TESTE"
20 STOP
30 RUN
```

Por RUN retornará 9/20 (código de STOP na linha 20).

Digite CONT e NEW LINE.

STR$

É uma função. Retorna um número dado na forma de STRING$.
Por exemplo:

```
 5 LET X$ = ""
10 INPUT X
15 LET X$ = X$ + STR$ X
20 PRINT X$
25 SCROLL
30 GOTO 10
```

PRINT X e/ou PRINT STR$ X produzirá o mesmo efeito.

TAB

É uma função de impressão. Desloca a posição de impressão n colunas à frente, onde o máximo de n é 255. Deve ser usado na forma:

```
PRINT TAB 10 ; "TESTE"
```

ou

```
PRINT "TESTE"; TAB 15; "TESTADO"
```

Veja PRINT, AT e a Memória de Vídeo.

TAN

Função. Fornece a tangente de um ângulo, informado em radianos.

Um radiano equivale a aproximadamente 57^{o}, ou mais, precisamente:

180^{o} = PI radianos

Programa para plotar a TAN:

```
10 FOR X = Ø TO 6 STEP .1
20 IF ABS (TAN X)< 2 THEN PLOT X*1Ø+2,
   INT (TAN(X)*1Ø+2Ø)
30 NEXT X
```

A restrição da linha 20 é devida ao fato da TAN de X tender ao infinito, quando X for um múltiplo ímpar de PI/2.

THEN

THEN é parte de uma instrução. Deve ser usada em conjunto com IF. (vide IF...THEN).

Na verdade, THEN apenas comuta o cursor no modo [L] para o modo [K], a fim de que possa ser inserida uma palavra-chave, mais especificamente , uma instrução a seguir da condicional proposta por IF... Por exemplo:

```
IF X>100 THEN GOTO 30
```

Experimente escrever:

```
1 REM REM
```

Aparentemente impossível. Digite então:

```
1 REM THEN REM
```

Retroceda o cursor e apague o THEN.

UNPLOT

É uma instrução. UNPLOT é geralmente usada após as instruções PLOT, porque seu único efeito é apagar uma posição anteriormente "plotada".

Assim como PLOT, UNPLOT vai de:

```
UNPLOT 0,0  a
UNPLOT 63,43
```

Lembre-se de que, para cada posição ou caractere de PRINT, existem 4 posições para PLOT.

Exemplo:

```
10 PRINT "DIGITE A DE 0 A 63"
15 INPUT A
20 PRINT "DIGITE B DE 0 A 43"
25 CLS
30 PLOT A,B
35 PAUSE 100
40 UNPLOT A,B
45 PAUSE 100
50 RUN
```

Veja PLOT

USR

É uma função para a chamada de sub-rotinas no código da máquina, no endereço dado(n), onde n deve ser um inteiro entre Ø e 65535. Número fora dessa faixa retorna código de erro B.

Para a utilização de USRn,é necessário ter-se,previamente, escrito a rotina em código, exceto, quando chamando as da ROM.

Como USR é uma função,não pode ser usada diretamente e sim como argumento de uma instrução. É válido:

```
RAND USR N
RUN  USR N
LET L=USR N
```

etc. ..., e por

```
PRINT USR N,
```

quando do retorno ao Basic, obtem-se impresso no vídeo o valor do register BC.

VAL

Função para a manipulação de STRING$.

Aplicando-se VAL a uma STRING, retorna uma expressão numérica.

A STRING usada como argumento de VAL deve conter somente caracteres numéricos e operadores aritméticos ou lógicos.

Por exemplo:

```
10 PRINT VAL "1+2+3"     resulta em 6.

10 LET X$ = "10"
20 LET Y$ = "3"
30 PRINT VAL (X$ + Y$)   retorna 13.
```

Operações aritméticas são também executáveis. Por exemplo:

```
10 PRINT VAL "SQR 9"   retorna 3

10 PRINT VAL "ABS-29"  retorna 29
```

Convertendo outros Basics

O "BASIC" é uma linguagem. Uma linguagem de programação e que, como toda linguagem, possui inúmeras variações, com poucas ou muitas diferenças, variando de computador para computador.
O "BASIC" criado pela Sinclair e utilizado pelos micros NEZ8000, CP200, TK82C, TK83 , TK85, Ringo e AS1000 é bastante semelhante ao "BASIC" "convencional", excluindo-se a manipulação de dados, apesar de seu funcionamento "sui generis" em relação aos demais "BASICs".

A seguir, apresentamos uma relação das principais diferenças entre os diferentes "BASICs" e o que poderíamos chamar de "BASIC ZX" ou "BASIC SINCLAIR".

NÚMERO DE INSTRUÇÕES POR LINHA

A maioria dos Basics permite o uso de várias instruções por linha, como por exemplo:

```
10 LET A=1:LET B=5:LET C=10:PRINT A,B,C
```

Neste caso, as instruções foram separadas por (:) dois pontos. No BASIC ZX só é permitido o uso de uma instrução por linha, como por exemplo:

```
10 LET A=1
20 LET B=5
30 LET C=10
40 PRINT A,B,C
```

AS VARIÁVEIS

A maioria dos BASICs permite a leitura e o envio das variáveis, subscritas ou não, aos periféricos, como o cassete, oferecendo grandes vantagens com o arquivamento de dados nos meios magnéticos.

No BASIC ZX, as variáveis são gravadas do e para o cassete, juntamente com o programa, ficando o arquivo de dados limitado à capacidade da RAM, menos o consumo do programa.

AS MATRIZES

Não requerem o uso de CLEAR e/ou DEF.
O BASIC ZX possui matrizes muldimensionais numéricas e alfanuméricas .Os subscritos se iniciam em 1.
Não existe A(0).

OS NOMES DAS VARIÁVEIS

No BASIC ZX as variáveis numéricas são de

nominadas por qualquer letra, seguida ou não de outros caracteres alfanuméricos. As variáveis alfanuméricas ou cadeias de caracteres são denominadas de A$ até Z$.

OS NOMES DAS VARIÁVEIS SUBSCRITAS

As matrizes numéricas podem ser denominadas de A a Z e as alfanuméricas de A$ a Z$, no BASIC ZX.

A INSTRUÇÃO LET

A maioria dos BASICs permite a omissão da palavra LET, como por exemplo:

```
10 A=1
```

no BASIC ZX:

```
10 LET A=1
```

AS INSTRUÇÕES INPUT e PRINT

Em alguns BASICs a instrução INPUT permite a entrada de vários valores simultaneamente, quando separados por vírgula. Exemplo:

```
10 INPUT A,L,P
```

Podendo ainda acumular o significado de PRINT, se na forma:

```
10 INPUT, "ALTURA,LARGURA,PESO",A,L,P
```

Esta única linha requer 6 linhas no BASIC ZX.

```
10 PRINT "ALTURA"
11 INPUT A
12 PRINT "LARGURA"
13 INPUT L
14 PRINT "PESO"
15 INPUT P
```

Na maioria dos BASICs, quando a instrução PRINT for ocasionar uma sobrecarga ou falta de espaço no vídeo, este é alterado automaticamente por uma função do tipo SCROLL, "abrindo" espaço para a nova mensagem. Este tipo de "proteção" não existe no BASIC ZX.

AS INSTRUÇÕES GOTO e GOSUB

Em alguns BASICs, é obrigatório que, a seguir das instruções GOTO e GOSUB, esteja especificado um número de linha e que esta exista.

Nestas instruções, o BASIC ZX é bem versátil. Não é obrigatório que a linha de número indicado exista. O programa reinicia-se

na primeira linha de número subsequente à indicada.
O BASIC ZX permite ainda que a definição do número da linha de destino seja feita por uma variável ou resultado de operações matemáticas. Exemplo:

```
10 GOTO X
10 X*A/2
10 GOSUB X*A/2
```

IF ... THEN

A maioria dos BASICs permite:

```
10 IF...THEN 100
```

No BASIC ZX é obrigatório o uso da expressão GOTO. Por exemplo:

```
10 IF...THEN GOTO 100
```

No BASIC ZX não é obrigatório que a seguir de IF...THEN seja especificado um número de linha, oferecendo as seguintes possibilidades:

```
10 IF...THEN PRINT "EUREKA"
10 IF...THEN LET X=X-1
10 IF...THEN RUN
```

Etc...

ON X GOTO, ON X GOSUB

Encontradas na maioria dos BASICs são instruções de desvio do programa que dependem do valor da variável X. Por exemplo:
1Ø ON X GOTO 1ØØ, 2ØØ, 3ØØ fará com que,se a variável X assumir o valor 1, o programa se desvie para a linha de número 1ØØ, se X assumir o valor 2 significa GOTO 2ØØ e se X = 3 GOTO 3ØØ .

No BASIC ZX, equivale a :

```
10 IF X=1 THEN GOTO 100
12 IF X=2 THEN GOTO 200
14 IF X=3 THEN GOTO 300
```

No BASIC Sinclair ou ZX, ON X não existe , mas poderíamos usar, por exemplo:

```
10 INPUT X
20 GOTO X * 100
```

DATA, READ, RESTORE

Encontradas na maioria dos BASICs, prestam-se à manipulação de dados. Exemplo:

```
10 DATA 13, 23, 33, 43
```

Esta instrução contém os valores 13, 23,33 e 43 que podem ser lidos ou transferidos para uma matriz com o uso de READ.Exemplo:

```
20 FOR N=0 TO 3
30 READ A(N)
40 NEXT N
```

Apresentamos,a seguir, tres versões para o BASIC ZX.

```
10 LET A(1) = 13
12 LET A(2) = 23
14 LET A(3) = 33
16 LET A(4) = 43
```

Esta solução custa 25 bytes ,por linha de programa. A solução abaixo é bem mais econômica em termos de consumo de memória.

```
1 REM XXXX (Obrigatoriamente comoprimeira linha).
POKE 16514,13
POKE 16515,23
POKE 16516,33
POKE 16517,43
12 FOR N = 1 TO 4
14 LET A(N) = PEEK(16513+N)
16 NEXT N
```

E esta outra, mais econômica ainda.

```
LET A (1) = 13
LET A (2) = 23
LET A (3) = 33
LET A (4) = 43
```

Use os comandos,diretamente, sem número de linha. Neste caso, RUN ou CLEAR apaga os valores armazenados.

RESTORE reinicia as definições das variáveis do início.

LEFT$, MID$, RIGHT$

Encontradas na maioria dos BASICs, são instruções para manipulação de cadeias de caracteres. Não são encontradas no BASIC ZX e podem ser substituídas, como a seguir:

LEFT$ (A$,X) fornece os primeiros X caracteres de A$.

Equivale a A$ (TO X)

RIGHT$ (A$,X) fornece os últimos X caracteres de A$.

Equivale a A$ ((LEN(A$)-X)TO).

MID$ (A$,A,B) fornece a quantidade B de caracteres da cadeia A$, a partir da posição A. Equivale a A$ (A TO (A+B)).

ASC, CHR$

A função ASC encontrada na maioria dos BASICs fornece o valor ou código do pri-

meiro caractere de uma cadeia de caracteres, de acordo com o ASCII.

ASC(A$) equivale a CODE (A$), no BASIC ZX. É importante notar que os códigos dos caracteres ZX não correspondem em nada ao ASCII, American Standard Code for Information Interchange, utilizado pela maioria dos computadores e periféricos da atualidade.

A função CHR$, no Basic ZX, não difere em nada .

PONTO FLUTUANTE

O BASIC ZX armazena números em cinco bytes com ponto flutuante. de $\pm 3 \times 10^{-39}$ a $\pm 7 \times 10^{38}$ com resolução de nove dígitos e meio.

OUTRAS DIFERENÇAS

Entre a maioria dos BASICs e o BASIC ZX existem outras diferenças, além das aqui relacionadas.

Contando os Bytes

De um modo geral, praticamente todas as instruções e funções custam apenas um byte. Como a maioria não pode ser usada isoladamente, é preciso considerar argumentos necessários ou aplicáveis a cada caso, para conhecermos o consumo de memória.

A seguir, apresentamos algumas possibilidades:

AS LINHAS DO PROGRAMA

Em uma linha do programa são gastos cinco bytes, além do seu conteúdo. Dois são para o número de linha, dois para arquivar o comprimento da linha em bytes e um para o término, dado por NEW LINE.

1 REM custa 6 bytes.

OS CARACTERES E PALAVRAS-CHAVE

Quando usados entre aspas, como parte de string, custam definitivamente apenas um byte.

OS SÍMBOLOS MATEMÁTICOS E DE PONTUAÇÃO

Vezes, mais, menos, divisão, potência, maior, menor, parêntese, ponto, ponto e vírgula, vírgula custam um byte. A vírgula, como ca

ractere de controle de impressão custa dezesseis bytes na memória de vídeo.

AS INSTRUÇÕES

CLS, CLEAR, CONT, COPY, FAST, SLOW, LLIST, LIST, REM, LPRINT, PRINT,RAND,RETURN,STOP, SCROLL, etc. ..., custam apenas um byte. Como linha de programa, custam seis bytes.

AS INSTRUÇÕES

LPRINT X, PRINT X, INPUT X, PRINT PI,PRINT RND, etc..., custam dois bytes. Como linha de programa, custam sete bytes.

AS INSTRUÇÕES

LPRINT X$,PRINT CHR$X, INPUT X$, etc...custam tres bytes. Como linha de programa, oito bytes.

AS INSTRUÇÕES LET, NA ÁREA DO PROGRAMA

LET X = A custa quatro bytes. LET X$ = "" custa seis bytes.

LET X = Ø custa dez bytes. LET X = 3+3 custa dezoito bytes. Como linha de programa custa vinte e três bytes.

AS INSTRUÇÕES PAUSE, GOTO, GOSUB

Custam dois bytes,se empregadas com variáveis como:

PAUSE X, GOTO X, GOSUB X

Custam,pelo menos, oito bytes, se empregadas com constantes numéricas como:

PAUSE 1, GOTO 1, GOSUB 1,

ou dez bytes

PAUSE 100, GOTO 999, GOSUB 100,

ou quinze bytes

10 PAUSE 100

AS INSTRUÇÕES POKE, PLOT, UNPLOT

POKE X,Y ou PLOT X,Y custa quatro bytes.No entanto, POKE 1,0 ou PLOT 1,0 custa dezesseis bytes.

10 POKE 16418,10 custa vinte e seis bytes.

A CONDICIONAL

IF A=1 THEN GOTO 1 custa dezenove bytes.
Possui sete caracteres, ou palavras-chave além de duas constantes numéricas que custam seis bytes cada.
10 IF A=100 THEN GOTO 9000 custa vinte e nove bytes.

A TABULAÇÃO TAB, AT

PRINT TAB X;N é a possibilidade que apresenta menor consumo de memória. Custa cinco bytes e como linha de programa dez bytes.

10 PRINT TAB 1; "X" custa dezoito bytes.

Cada posição de impressão ou coluna deslocada pela função TAB irá custar um byte extra na memória de vídeo.

10 PRINT AT 0,0; "X" custa vinte e seis bytes.

Cada posição de impressão deslocada irá custar um byte extra na memória de vídeo.

AS FUNÇÕES

ACS, ASIN, ABS, ATN, COS,SQR,INT, LN, TAN, etc. ..., custam um byte, mas, como não são válidas sem argumento, custam "caro".

LOOPING

10 FOR N=1 TO 9 custa vinte e tres bytes.
10 FOR N=1 TO 1000 custa vinte e seis bytes.
30 NEXT N custa sete bytes.
10 FOR N=1 TO 1000 STEP 2 custa trinta e quatro bytes.

SUBSTRING$

10 PRINT A$ custa oito bytes.
10 PRINT A$ (TO 6) custa dezoito bytes.
10 PRINT A$ (TO 20) custa dezenove bytes.
10 PRINT A$ (2TO20)custa vinte e seis bytes.

AS INSTRUÇÕES DIM, NA ÁREA DO PROGRAMA

10 DIM A(X) custa dez bytes.
10 DIM A(X,Y) custa doze bytes.
10 DIM A(3) custa dezesseis bytes.
10 DIM A$(3) custa dezessete bytes.
10 DIM A(100) custa dezoito bytes.
10 DIM A$(100) custa dezenove bytes
10 DIM A(3,3) custa vinte e quatro bytes.

O consumo de memória, citado acima, não inclui a área reservada para os elementos da matriz. Vide abaixo.

AS VARIÁVEIS, NA ÁREA DAS VARIÁVEIS

As NUMÉRICAS custam cinco bytes, mais o número de caracteres do nome. Exemplo:

LET A=1 custa seis bytes e LET AA=1 custa sete bytes.

As ALFANUMÉRICAS custam tres bytes, mais a quantidade de caracteres empregados.

De CONTROLE , como FOR X-NEXT X custam dezoito bytes.

AS VARIÁVEIS SUBSCRITAS, NA ÁREA DAS VARIÁVEIS

MATRIZES NUMÉRICAS custam quatro bytes, mais duas vezes a quantidade de dimensões, mais cinco vezes o número de elementos.
Exemplos:

A (100) custa 4 + 2*1 + 5*100 = 506 bytes.
A(100,7) custa 4 + 2*2 + 5*100*7=3510 bytes.
A(100,5,7) custa 4 + 2*3 + 5*100*5*7=17510 bytes.

MATRIZES ALFANUMÉRICAS custam quatro bytes mais duas vezes o número de dimensões, mais a quantidade de caracteres. Exemplo:

A$ (100) custa 4 + 2*1 + 100 = 106 bytes.
A$(5,100) custa 4 + 2*2 + 5*100=508 bytes.
A$(100,5,7)custa 4 + 2*3 + 100*5*7 =3510 bytes.

Economizando Memória

01. Elimine as instruções REM, prefácios explicativos e linhas que excluam respostas indesejáveis.

02. Imprima todas as mensagens e símbolos, no canto esquerdo da tela.

03. Reduza o número de linhas do programa, ao mínimo. Cada linha de programa custa cinco bytes. Dois para armazenar o número de linha, dois para o comprimento e um para o término (New Line).

04. Não dimensione matrizes, a menos que necessário.

05. Elimine os resultados intermediários, em cálculos. Por exemplo: A/(B*C), use A/B/C.

06. Reduza a uma mesma linha todas as operações aritméticas, relacionais e lógicas possíveis.

07. Remova IF X = 0 THEN ... e use IF NOT X THEN ..., você economiza uma constante numérica, seis bytes.

08. Remova IF T > 100 THEN LET T=T + 1 e use LET T = T + (T > 100). Na sentença matemática se verdadeiro =1, se falso = 0.

Remova

```
100 IF INKEY$ = "M" THEN LET Y=Y+1
110 IF INKEY$ = "N" THEN LET Y=Y-1
```

e use

```
100 LET Y=Y+(1 AND INKEY$ ="M")-(1 AND
    INKEY$ ="N")
```

ou

```
100 LET Y=Y +(INKEY$ = "M") -(INKEY$ =
    "N")
```

pelo mesmo princípio do ítem 08.

09. Use variáveis, ao invés de constantes numéricas, sempre que for repetir um número, mais que tres vezes, no programa. Cada constante numérica custa mais seis bytes, além dos caracteres mostrados no vídeo. Isto porque são sempre seguidas do código 126 e outros cinco bytes para o valor numérico em si. Isto ocorre da mesma forma com valores, como 0, 1 ou 1.3233.

10. Use os valores das funções e códigos disponíveis, ao invés de introduzir constantes numéricas e/ou variáveis. Exemplo: LET A=0 ou LET A=1 ocupa dez bytes.
Use:
LET A=PI/PI, ou LET A=PI-PI, etc...

11. Remova 10 LET A=1, 20 LET B=1 e use:
    10 LET A=1, 20 LET B=A.
    Isto economiza seis bytes.

12. Dê preferência a PEEK e POKE, para armazenar valores, sempre que possível.

13. Transforme as constantes numéricas , se forem muitas, em uma só variável literal (alfanumérica), isto é,coloque todos os caracteres numéricos um ao lado do outro, atribua-lhes o nome de uma variável literal e passe a manipulá-los,como tal. Utilize CODE A$ e/ou LET A$(X TO N), para recuperar os valores.

14. "Overlaying". Você pode sobrepor um programa em outro de várias formas, com inúmeras vantagens. Por exemplo: use um pequeno programa só para definir as variáveis de um programa principal, como abaixo:

```
10 LET A=1
12 LET B=2.2
14 LET SALÁRIO=23568
16 LET C=12
```

Rode"a definição"das variáveis e a seguir apague-a, linha por linha. Os valores permanecerão arquivados,se você não usar RUN ou CLEAR. Use GOTO 1.

15. Quando arquivando dados, use sempre que possível, matrizes alfanuméricas ao invés de numéricas.

    Por exemplo: um gabarito de provas com 100 questões e 5 alternativas de resposta poderia ser arquivado em uma matriz do tipo X(100), sendo que DIM X (100), iria custar 506 bytes.
    Usando X$(100), além das 5 alternativas de resposta, numéricas, poderíamos ter respostas A,B,C,D ou E e a matriz custaria apenas 106 bytes.

Veja o capítulo "Contando os bytes".

PUBLICAÇÕES *MICRON ELETRÔNICA*

PARA COMPUTADORES

sinclair®

SINCLAIR É MARCA REGISTRADA DE SINCLAIR RESEARCH LTD.

• DICIONÁRIO DO BASIC SINCLAIR

Para os micros ZX81, NEZ8000, CP200, TK82C e TK83, TK85, RINGO, AS 1000, etc... Reune todas as Instruções - Funções - Operadores e Caracteres descritos e exemplificados em ordem alfabética .
Inclui Divisão do Basic Sinclair - Convertendo outros Basics -Contando os Bytes - Economizando Memória.

• 200 DESENHOS PARA TK E CP200

. Mais de 200 desenhos para jogos e ilustrações, codificados.
. Programas para : edição dos desenhos codificados; cópia instantânea do Display, simulando movimento; transposição de vídeo entre programas, etc....
. Ensina as técnicas de vídeo, mapeamento, movimento, etc...

• 45 PROGRAMAS PRONTOS PARA RODAR EM TK82C E NE Z8000

Arquivos - Estoque - Plano Contábil - Folha de Pagamento - Agenda Telefônica - Caça ao Pato - Trilha - Jogo da Velha - Forca - Dado Tabelas - Tabuadas - Conversão de Coordenadas - Média - Progressão - Tabela Price - Fibonacci - Depreciação - Renumerador de Linhas em Código - Etc....

• APLICAÇÕES SÉRIAS PARA TK85 E CP 200

Quem é Sinclair - Convertendo outros Basics - Contando os Bytes Economizando Memória - Fluxogramas - Top Down - Erros da ROM - Conhecendo a Impressora - Chaining Programas - Sub-Rotinas em Cassete - Folha de pagamento - Balancete - Correção Monetária do Imobilizado - Das Contribuições do IAPAS - Contas a Receber - Cadastro de Clientes - Conta Bancária - Correção de Provas - Processador de Textos - Estatística - Custos - Orçamento Doméstico - Ramtoper em Código - etc. ...

• 30 JOGOS PARA TK82 E CP 200

Damas - Labirinto - Enterprise - Golfe - Velha - Visita ao Castelo Cassino - Roleta Russa - Corrida de Cavalos - Vinte e Um - Cubo Mágico - Senha - Banco Imobiliário - Forca - Dados - Invasores-etc.

PROGRAMAS NO CÓDIGO DA MÁQUINA

Inversão de Vídeo - Som por Software - Labirinto - Destrava Soft

• CÓDIGO DE MÁQUINA PARA TK E CP 200

Números Binários e Hexadecimais - Arquitetura do Z80 - Editando em Código - Programa para Edição - As Instruções do Z80 em Exemplos - Sub-rotinas da ROM - A ROM de 8K - Dicionário das Instruções - HEX X Mnemônicos - Hex X Decimal - Incluindo os Programas : Scroll - Save Display no Ram Top - Contadores de Tempo ou Ponto - DataFile - Renumber - Labirinto - Som por Software - Micron Pac - Bombardeio, etc...

ZX SOFTWARE

OS QUATRO MELHORES PROGRAMAS EM CÓDIGO, JAMAIS PRODUZIDOS PARA UM SINCLAIR EM UM ÚNICO CASSETTE. POR APENAS Cr$ 15.000

Assembler

Escreva os seus programas em código, usando diretamente os mnemônicos do Z80 com este Editor e Monitor de Assembly. Interpreta todas as instruções do Z80. Oferece excelentes facilidades de edição, manipulação do cursor, códigos de erro, entrada de texto, números, labels, repetição de tecla, etc...

Disassembler

Lê códigos no Assembly do Z80. Fornece os endereços em decimal, os códigos em hexadecimal seguidos dos mnemônicos completos. Interpreta todas as instruções do Z80.

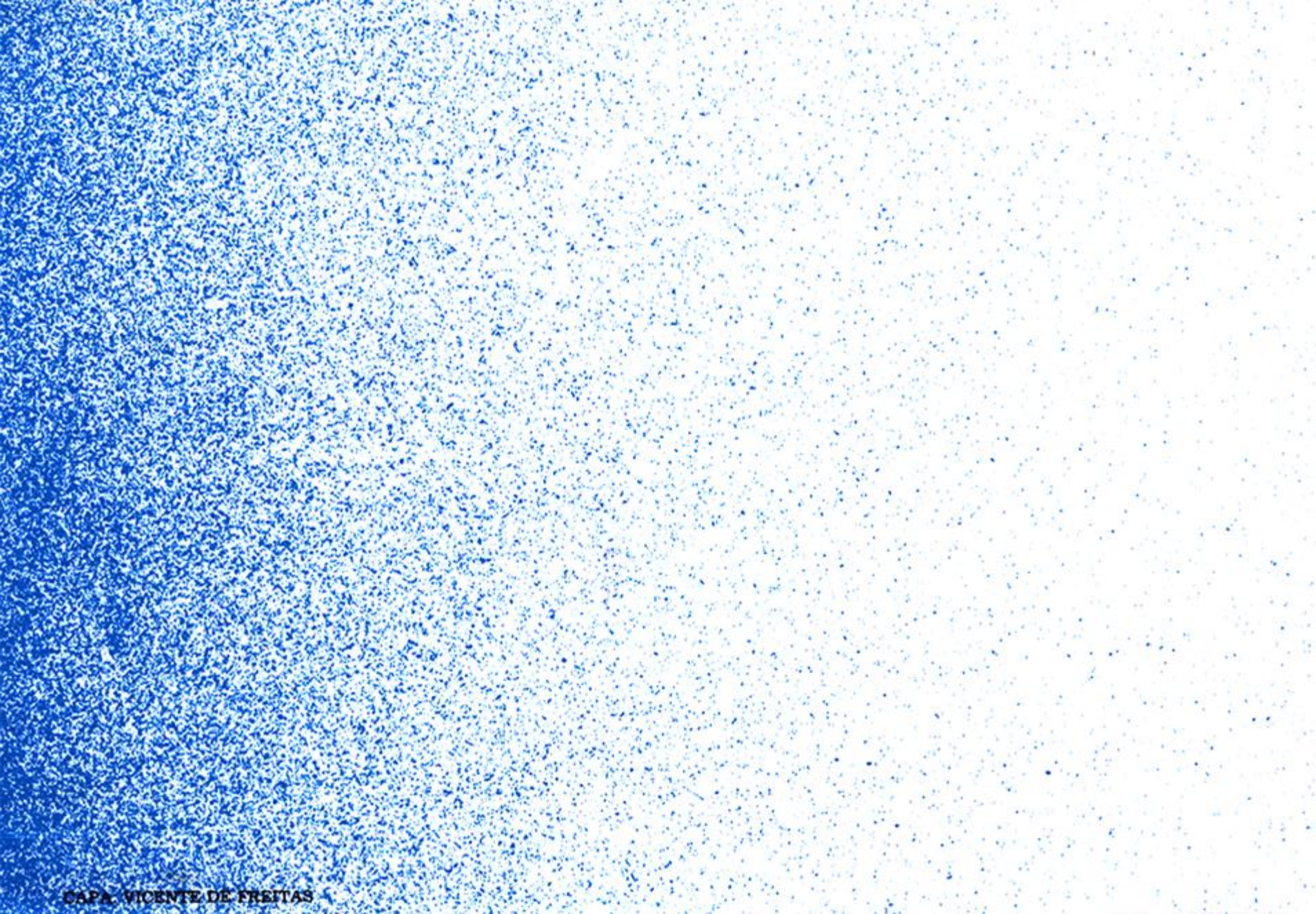
Compiler

Transforme instantaneamente os seus programas em Basic em programas em código, usando o Compiler. Aceita quase todos os comandos do Basic Sinclair e passa a rodar os programas até 50 vezes mais rápido.

Monitor

Programa para estudo de programas em código. Permite listar a Memória, Registers e Flags. Pode-se ainda introduzir Breakpoints, converter Hex para Decimal, decimal para Hex etc...

Distribuído por : *MICRON ELETRÔNICA*



CAPA: VICENTE DE FREITAS